Patent Application

for

DATA PROCESSING APPARATUS

Inventor(s):

GEORGES EMMANUEL BLIN
LE HUAN TRAN

Cross-Reference to Related Applications

[0001] This application claims the benefit under 35 U.S.C. §119 of the following co-pending and commonly assigned foreign patent application, which application is incorporated by reference herein:

[0002] United Kingdom Patent Application No. 02 16 825.0, entitled "DATA PROCESSING APAPRATUS", filed on July 19, 2002.

Background of the Invention

1. Field of the Invention

[0003] The present invention relates to data processing apparatus and a method of processing data.

2. Description of the Related Art

[0004] Data processing systems for editing and manipulating large amounts of data, for example image clips, have been known for a number of years. Such systems contain large data storage devices, typically in the form of a RAID (redundant array of independent disks).

[0005] A particular piece of data may be very large, for example a clip which is to be used in an editing process may comprise many megabytes of data. Therefore, if an attempt is made to load such data onto a RAID, and the available storage space on the RAID is insufficient, processing time and operator time is wasted. It is therefore known for systems to have a process for measuring the available storage space before the storage process is commenced. However, such measuring processes themselves require a substantial processing effort, and the corresponding processing time can impact on the workflow of

the human operator.

Brief Description of the Several Views of the Drawings

[0006] *Figure 1* shows a system **100** for editing data;

[0007] *Figure 2* shows computer **101** of *Figure 1*;

[0008] *Figure 3* shows framestore **105** of *Figure 1*;

[0009] *Figure 4* shows an illustration of a striping process for storing image frames;

[0010] *Figure 5* shows an illustration of three partitions **501**, **502** and **503** co-existing on the RAID **105**;

[0011] *Figure 6* shows an example of a bitmap **601** representing a partition;

[0012] *Figure 7* illustrates eight data caches **701** to **708**;

[0013] *Figure 8* shows a flow chart illustrating the operation of the system **100**;

[0014] *Figure 9* shows the step **803** of acting upon a request to load image frames onto the framestore;

[0015] *Figure 10* shows the step **901** of determining available RAID space in further detail;

[0016] *Figure 11* shows the step **806** of acting upon a request to delete image frames from the framestore **105**.

Detailed Description of the Invention

[0017] An embodiment of the invention will now be described by way of example only with reference to the previously identified drawings.

[0018] In this embodiment the data being stored is in the form of clips of frames, and so in this example data processing apparatus is provided by system **100** for editing image data, as illustrated in *Figure 1*. The system is used by a human operator for editing,

modifying, processing and adjusting video or film image data to form an output sequence that will eventually be stored onto digital tape. In other embodiments the data stored may be, for example, financial data, customer databases or any other type of data, and the processing system would vary accordingly.

[0019] The system comprises a computer **101** that in this example is an Octane manufactured by Silicon Graphics Inc., a monitor **102**, a graphics tablet **103** to allow the user to interact with a graphical user interface presented by the monitor and a keyboard **104** to facilitate alphanumeric input.

[0020] The system **100** further comprises data storage means. In this embodiment the storage means is a disk-based frame storage system, referred to herein as a framestore **105**, and in preparation for image editing, images from one or more film or video clips are transferred to the framestore **105** from a digital tape player (not shown). The digital tape player may be local to the system **100** or remote, with the transfer taking place via a network. The framestore **105** comprises several high capacity hard disk drives configured as a RAID (redundant array of independent disks), arranged to supply and store image data in parallel across several individual drives at once.

[0021] Using the framestore **105**, it is possible to play back and record high resolution film images or video images at any location in a clip without having to wait for a tape wind mechanism to reach the required frame, thereby facilitating non-linear editing. Furthermore the framestore allows real time play and record of image data.

[0022] The computer **101** includes a CD-ROM drive **106**, allowing program instructions to be loaded onto a hard disk within the Octane, from a CD-ROM **107**.

Figure 2

[0023] Computer **101** of *Figure 1* is detailed in *Figure 2*. The computer includes processing means, provided in this example by one or more programmable processing devices **201** that communicate with system memory **202**, a local hard disk drive **203**, a first interface **204** for communicating with the framestore **105** and a second interface **205** for communicating with the keyboard **104**, touch tablet **103** and stylus.

[0024] Processing device **201** operates in response to program instructions read from system memory **202**. On initiation, program instructions are loaded into the system memory **202** from the local disk **203**. Local disk **203** receives program instructions via data storage media such as a CD-ROM **107** receivable within a CD-ROM reader **106**.

[0025] The system memory **202** contains kernel memory **206** which is reserved for the use of the operating system for computer **101** and also for data storage as herein described.

[0026] It will be appreciated that computer **101** is merely an example of a way in which processing means and memory means can be provided within data processing apparatus. Any computer capable of carrying out complex instructions and communicating with a data storage means, whether internal or external, would be suitable.

Figure 3

[0027] Framestore **105** is detailed in *Figure 3*. In this example the array has a total of four magnetic disk drives **301** to **304**, each with a storage capacity of four hundred gigabytes. Image data is received from the processing system **101** and is supplied to the

processing system **101** over a SCSI interface **305** or, alternatively, a fibre channel interface (not shown). Interface **305** communicates with a SCSI controller which in turn communicates with the individual drives over a channel **306**.

[0028] The individual frames stored on the framestore **105** form contiguous clips, usually derived from computer animation systems, video sources or cinematographic film sources. The frames are arranged to be displayed at a particular display rate, such as thirty frames per second for NTSC, twenty-five frames per second for PAL or twenty-four frames per second for cinematographic film. Framestore **105** is therefore configured to allow these different types of frames to be transmitted at display rate or at multiples of display rate.

[0029] In addition to being displayed at different rates, and therefore requiring differing data transfer rates, the actual size of the frames also varies for different frame formats. Thus, for example, a frame of NTSC video or PAL video requires approximately one megabyte of storage space. High definition television systems require an ever-greater degree of storage capability per frame and systems capable of processing images derived from cinematographic film may yet require a greater degree of storage per frame. The system therefore needs to be configured to allow frames to be transported at selected display rates and at selected frame definitions.

[0030] The framestore is optimised by dividing each image frame into a plurality of stripes and then writing each stripe to an individual disk. In this way, data defining an image frame is distributed over three of the disks **301** to **303**. In addition, a further disk **304** is required for parity data where similar bits within each stripe are XORed together to produce a parity stream that is written to the redundant disk. In this way, the loss of

data from any one disk may be reconstituted by performing the XORing process for all the remaining data. Further details of such a system are given in the present Assignee's United States Patent No. 6,118,931.

Figure 4

[0031] A striping process for storing image frames is illustrated in *Figure 4*. An incoming frame **401** is divided into three stripes, identified as stripe zero, stripe one, and stripe two. A storage control process **402** on framestore **105** performs an XOR operation to generate parity data and thereafter writes the data in parallel to disks **301** to **304**. Thus in this example, disk **301** receives data from stripe two, disk **302** receives data from stripe one, disk **303** receives data from stripe zero, disk **304** receives the parity data. The addressing of data from the stripes may identify substantially similar locations but with the application of an appropriate off-set. Thus, data is read from stripe one at the same locations as data being read from stripe zero but with an appropriate off-set as identified by arrow **403**.

[0032] Having established a system of using four disks to stripe image frames as shown in *Figure 4*, applications executed by processing system **101** may access the storage means, but from the perspective of processor **201** on computer **101** the four grouped drives operate as a single logical volume.

[0033] As mentioned above, the framestore **105** is used to process image frames of various predetermined definitions and display rates. In each case, a striping process similar to that described above may be used, resulting in each frame being stored as one or more stripes on each of the disks **301** to **304**. However, due to the varying quantity of data required to define images of varying definition, the lengths of the stored stripes

depend upon the definition of the respective images. For example, both NTSC and PAL images may be stored using the above described process, but the NTSC images will require the storage of slightly shorter stripes than for the PAL images. Consequently, in order to store the data efficiently, image frames are stored in separate partitions on the RAID, with each partition configured to accept image data relating to image frames of a predetermined definition.

[0034] Higher definition images may also be stored using the striping process described above. However, if the image definition is too large for image data to be written to and read from the framestore at the required rate, one or more additional RAIDs similar to framestore **105** may be used. In such a case, the RAIDs are used in conjunction, such that each frame is striped across the disks of each array.

Figure 5

[0035] An illustration of three partitions **501**, **502** and **503** co-existing on the framestore **105** is shown in *Figure 5*. Each of the four disks **301** to **304** is similarly partitioned and configured to store image frames having three different defined definitions A, B and C. Consequently, partition **501** has storage elements A0, A1, A2 etc. of a first size, chosen for the efficient storage of stripes from frames of definition A, partition **502** has storage elements B0, B1, B2 etc. configured to receive stripes from frames of definition B, and partition **503** has storage elements C0, C1, C2 etc. configured to receive stripes from frames of definition C. Thus, on a request to store image frames of a particular one of said definitions, the storage process **402** writes data into the respective partition.

[0036] It should be understood that each partition, such as partition **501**, extends

across all disks in the framestore **105** and a data storage element, such as element A10 in partition **501**, comprises four storage blocks, one on each of the four disks, with each block being at the corresponding location on each disk.

[0037] In an alternative embodiment, in which the framestore comprises two or more such RAIDs used in conjunction, a partition may extend across all disks in all of the RAIDS, or a group of disks. Thus, for example lower resolution image frames may be stored on one four-disk RAID while higher definition frames are stored on the eight disks of two RAIDs. In this embodiment, each of the partitions contains a certain number of storage elements, but the elements from some partitions comprise blocks from four disks while storage elements of the high definition partitions may contain blocks from eight or more disks. In further alternative embodiments the data may be stored on framestore **105** in other ways, for example without partitioning.

[0038] It will be appreciated that the example given here of a framestore comprising one or more RAID is a way of storing data that is particularly appropriate to image data. RAIDs are also suited to other types of large amounts of data but, dependent upon the type of data stored, the storage means may vary. For example, it may be the hard disk drive of a computer or another type of external storage device.

Figure 6

[0039] In order to manage the storage of image frames to the framestore **105** the processor **201** utilises usage data which is stored within kernel memory **206**. The usage data stores information on whether each storage element in framestore **105** is being used or not. In this embodiment the usage data takes the form of one or more bitmaps, which are arranged to represent the status of the storage elements within the partitions of the

RAID. Each bit within the bitmap corresponds to just one of the storage elements, and is arranged to indicate whether or not a storage element is currently being used to store image data. Consequently, if a particular storage element is not currently being used, the corresponding bit is set to zero, and if it is being used it is set to one. In order to maintain the accuracy of the bitmap data, the processor **201** updates the bitmap as image data is written to and deleted from the framestore **105**.

[0040] An example of a bitmap **601** representing a partition is shown in *Figure 6*. The bitmap **601** may, for example, represent partition **501** of *Figure 5*. A small section **602** of the bitmap **601** is shown as enlarged portion **603**. As illustrated by the enlarged portion **603**, the bitmap comprises ones and zeroes, indicating the current use or non-use respectively, of the corresponding storage element within partition **501**. In this example a storage element is the space allocated to the storage of a single frame, but it will be appreciated that there are many ways of dividing up storage space and that the exact nature of a storage element will be dependent upon the type of data and the way in which it is stored. For example, when storing more traditional types of data a storage element might be a sector.

Figure 7

[0041] A human operator of system **100** may need to process image data from many different clips, each comprising frames of one of several different predetermined image definitions. In order to edit, manipulate, etc. said clips, they are first stored onto the framestore **105**.

[0042] Thus, during an editing process the processor **201** receives a plurality of requests each requesting that that a clip comprising image data is to be stored on

framestore **105**. These requests may originate from an application run by computer **101**, or in alternative embodiments from other computers connected to system **100**. If the processor **201** merely passed the data to the storage control process **402** for storage, there would be a possibility that the available storage space would be filled before all data defining the clip were stored, and thus the storage process would be unsuccessful.

[0043] Therefore, before commencing the storage of a clip, the processor **201** determines whether the framestore **105** contains sufficient free storage space to store said clip. This may be done by parsing the relevant bitmap to determine how many frames of the required image definition could be stored in the partition. If it is determined that sufficient storage space exists then the clip is stored. If sufficient storage space does not exist then an appropriate message may be displayed to the user.

[0044] However, due to the large storage capacity of the framestore **105** and the correspondingly large size of the bitmap or bitmaps, the parsing process may take as much as two or three seconds.

[0045] In order to improve workflow of the system **100**, the kernel memory **206** further includes eight data caches **701** to **708** inclusive, as illustrated in *Figure 7*. Each of the data caches is configured to receive information regarding a frame definition, or size, and the corresponding number of storage elements which are currently available. Thus, for example, cache **701** may contain a frame definition, or size, of 1000, indicating NTSC frames and 10890, indicating that 10890 storage elements are currently not being used.

[0046] After parsing a bitmap as described above, the processor **201** stores the result to one of the caches. Consequently, if the storage request cannot be serviced, due to lack

of storage space, the cached data becomes of use to further storage requests. That is, if a further request is received to store a clip of the same frame definition then, instead of parsing the bitmap, the necessary information can be read immediately from the cache, and a few seconds may be saved in processing time and human operator's time.

[0047] Thus the caches **701** to **708** provide a datastore which, when used, contain data indicating the number of image frames of a predetermined definition which may be received by non-used locations of the framestore.

[0048] It will be apparent to the skilled user that the exact nature of the datastore is dependent upon the nature of the storage elements and usage data. For example, when storing financial data it would not appropriate to have different caches for frame resolutions. It may be that a single cache is appropriate. If more than one cache is necessary then each should be configured to receive information identifying the type of data stored.

Figure 8

[0049] A flow chart providing a simplified illustration of the operation of the system **100** is shown in *Figure 8*.

[0050] Following power on at step **801**, the system is initialised at step **802**. During the initialisation, the operating system is booted up, hardware drivers are initialised, the editing application program is started, and bitmaps representing storage usage within the framestore **105** and the corresponding caches are initialised.

[0051] At step **803** a question is asked to determine whether a request has been

received by processor **201** to load image frame data onto the framestore **105**. If this question is answered in the affirmative then the request is acted upon at step **804**, before step **805** is entered. A second question is asked at step **805**, to determine whether a request has been received to delete image frames from the framestore. If this question is answered in the affirmative then the request is acted upon at step **806** before step **807** is entered.

[0052] At step **807** image data is edited, manipulated etc. in response to input signals received from the keyboard **104**, or graphics tablet **103**. At step **808** it is determined whether or not a request has been received, indicating that the editing session is to be terminated. If this question is answered in the affirmative then the system may be shut down at step **809**, otherwise steps **803** to **808** are repeated.

[0053] Of course, the order in which steps **803** to **807** are performed may be varied from the flow chart of *Figure 8*, which is merely intended to provide a simple example.

[0054] In this embodiment the storage control process **402** is contained within framestore **105**, meaning that processor **201** receives requests to store data and, after determining whether there is enough space to store the data, it then passes the request and the data to storage control process **402.** This process is responsible for the actual storage of the data, for example determining the location of available space and recording the addresses of stored data. A similar process occurs with delete requests. However other RAIDs and, indeed, other external storage systems do not include a storage control process. This is also true if the data storage means is an internal hard drive. In these cases it will be understood that the processor takes on the role of the storage control process. Additionally, in this embodiment the datastore **206** is contained

within system memory **202** and is therefore maintained and interrogated by the processor **201**, either by requesting the necessary information from storage control process **402** or by keeping track itself of how much information has been stored, as will be described further below. However the memory means could be provided within framestore **105**, in which case it would be more appropriate for storage control process **402** to be responsible for updating the datastore. This would be particularly suitable when a storage means is accessed by more than one computer.

Figure 9

[0055] The step **803** of acting upon a request to load image frames onto the framestore is shown in more detail in *Figure 9*. At step **901**, processor **201** determines the available free storage space on the framestore **105** for storing image frames of the required definition. At step **902** a question is asked to determine whether the number of frames which may be stored, as determined at step **901**, is greater than or equal to the number of frames requested to be stored. If it is determined at step **902** that the framestore has insufficient storage space then an appropriate message is displayed on monitor **102** and step **804** is completed.

[0056] Alternatively, if step **902** finds that there is sufficient storage space, then at step **903** image frames are stored to the appropriate partition of the framestore **105** in accordance with the request. The relevant bitmap is updated at step **904** to reflect the changes made at step **903** to the storage space usage. At step **905** the corresponding cache **701**, **702**, **703**, **704**, **705**, **706**, **707** or **708** is flushed, and the flushed cache is then updated at step **906**, to complete step **804**.

[0057] Step **906** of updating the cache may be performed by re-parsing the relevant

bitmap. Alternatively, the new cached value may be calculated from the flushed value and the number of frames stored at step **903**.

Figure 10

[0058] The step **901** of determining available framestore space is shown in further detail in *Figure 10*. At step **1001** a question is asked to determine whether the available framestore space for image frames of the required definition has already been calculated and stored within one of the eight caches. If so, then the required information is read from the relevant cache at step **1004**, and step **901** is completed. Otherwise, if the question asked at step **1001** is answered in the negative, then the appropriate bitmap is parsed at step **1002** to determine how many storage elements are presently not being used, and hence how many frames of the requested image definition may be stored. The frame definition and number of frames which may be stored is then cached at step **1003** to complete step **901**.

Figure 11

[0059] The step **806** of acting upon a request to delete image frames from the framestore **105** is shown in more detail in *Figure 11*. At step **1101** frames are deleted from the relevant partition of the framestore **105** by storage control process **402**, in accordance with the request received at step **805**. At step **1104** the bitmap corresponding to said partition is updated by processor **201**, and then the corresponding cache is flushed and updated at steps **1105** and **1106** respectively. Thus steps **1104** to **1106** are, in essence, the same as steps **904** to **906**.